Steganography in TCP & IP headers

Daniel Ramírez

Resumen – La esteganografia es la disciplina en la que se estudian y aplican técnicas que permiten el ocultamiento de mensajes u objetos, dentro de otros, llamados portadores, de modo que no se perciba su existencia.Durante este TFG, intentaremos demostrar que es posible ocultar información dentro de la cabecera TCP (nivel de transporte) y la cabecera IP (nivel de red). La principal motivación para intentar demostrar que es posible ocultar información, es el hecho de poder enviar información sin que sea posible detectarla utilizando algunos campos en las cabeceras TCP e IP.Gracias a la implementación hemos podido demostrar que es posible modificar y ocultar información dentro de las cabeceras. El desarrollo y la implementación han sido posible usando Python y usando un Framework de Python llamado Scapy.

Palabras claves – Esteganografia, ocultación de la información, TCP, IP

Abstract – Steganography is the discipline in which is studied and applied techniques for hiding messages or objects, within others, called carriers, so that no notice of its existence.During the TFG, it demonstrates that it is possible to hide information within the TCP (transport layer) header and IP header (network layer). The main motivation to try to prove that it is possible to hide information is being able to send information without it being possible to detect using some fields in the TCP and IP headers. Thanks to the implementation we have proven that it is possible to modify and hide information in the headers. The development and implementation has been possible using Python and using a Python Framework called Scapy.

Index Terms - Steganography, information concealment, TCP, IP

Contact E-mail: daniel.ramirezm@e-campus.uab.cat
Specialization: Tecnologies de la Informació.
Project mentor: Guillermo Navarro (dEIC)
Year 2014/15

1. Introduction

THE utility of using steganography is manifested in the so-called problem of the prisoner.

Briefly, into the problem of the prisoner, we have two prisoners, A and B, who want to communicate confidentially to escape. The problem is that it can only exchange messages through a guardian, W. The guardian can read, modify or generate messages himself. If the guardian detects any communication that can be used to escape (e.g. Detects encryption) will stop transmitting messages. In this scenario the prisoners need to establish a covert channel.

The use of steganography allows to have a covert channel so that we can communicate without being detected. The strategy followed steganography to solve the problem of the prisoner is to hide data that does not want them to be detectedbetween the messages specified by the guardian.

Starting from this belief, we present this project that aims to fulfill all the requirements for an application to allow a certain individual, group or community to communicate using a cover channel that allow us to and receive data only between the sender and receiver.

But before sending data we have to find a good covert channel inside the TCP and IP header and the most important thing is have access to the transport and network layer for manipulate the header and send our own packet.

1.1 Personal motivation

I have chosen this project to develop because I think it is very important in the field of computer security issue, being able to send information that only the sender and receiver without a third party to manipulate or read the information.

2. State of the Art

Steganography has been around since before the fifteenth century and it has been in constant evolution. Examples of early steganographic technique are such as writing with "invisible ink" made of vinegar, lemon juice, milk, etc.

The steganographic techniques have evolved in line with technological development, as well as during World War II. The Newspapers for sending hidden signals were used by performing marks in certain letters, although by themselves they went unnoticed were transmitting information set.

This example in which usually the steganography has been more present, and as you can imagine are on the military environment or intelligence agencies among others.

The evolution of steganography techniques became more sublime across the ages, but the main principle remained the same.

Every secret message is carried within some other entity and thus communication can remain clandestine.

Today, when we speak of steganography we refer mainly to the concealment of information within an environment of digital channels: communication protocols, executable files, text documents, digital audio, images, etc. In most cases, the container file is known but what is ignored is the algorithm or technique of inserting information into said container object.

In recent times, steganography has gained great interest because these techniques are presumed to have been and are used with very different purposes by terrorist organizations and criminals, and are closely related to anonymity techniques such as voting or electronic money among others.

2.1. Network Steganography principles

While cryptography protects messages from being captured by unauthorized parties, steganography techniques enable concealment of the fact that a message is being sent, and, if not detected, make the sender and the receiver "invisible". Thus steganography potentially provides not only security, but also anonymity and privacy, which become understandable desires in modern societies which force us to take part in an increasingly intensive and complex social relations.

Obviously, the anonymity potential of steganography, while can be considered as beneficial in the context of protecting privacy, adds new type of threats to individuals, societies and states. The tradeoff between the benefits and threats involves many complex ethical, legal and technological issues. Here we consider the latter in the context of communication networks.

Steganography as a network threat was marginalized for few years but now not only security staff but even business and consulting firms are becoming continuously aware of the potential danger and possibilities it creates

In order to minimize the potential threat to public security, identification of such methods is important as is the development of effective detection (steganalysis) methods.

This requires both an in-depth understanding of the functionality of network protocols and the ways in which it can be used for steganography.

3. Objectives

In this items for specify the objectives of this project we have to difference between the primary objectives and the secondary:

3.1. Primary objectives

- Study steganography techniques in existing networks.
- Study in which fields of the TCP header can be used as a covert channel.
- Development a steganography techniques in network protocol IPv4.
- Implementation of the developed techniques using Scapy a Python Framework.
- Capture and analyze our TCP packets.
- Export the captured packets and recover the original message.

3.2. Secondary objective

- Study in which fields of the IP header can be used as a covert channel.
- Development a steganography techniques in network protocol IPv4.
- Implementation of the developed techniques using Scapy a Python Framework.
- Capture and analyze our IP packets.
- Export the captured packets and recover the original message.

4. Methodology

We can divide the methodology used in two parts, the methodology for the research and development of the method hide and the methodology used to implementing.
We will begin by explaining the methodology of development.
When it comes to perform research objectives and development of the project, was followed by a method of three phases:

1) Study and preliminary search of information about the topic in particular, to acquire knowledge
necessary to carry out phase development.

2) Theoretical development of the techniques implemented, based on the information acquired in phase 1 and the contrasting result this information to make sure that there are no errors.

3) Finally, perform a decision on the technique to implement.

Up to this point the methodology used for the research and development. Now we will explain the methodology implementing:

1) First, we perform an analysis of requirements on the application design.

2) Implementation of the application.

3) Phase of testing of the application, to confirm the proper working.

4.1. Project planning

The project was intended to be developed over the course of 13 weeks from its start on October 2014.
Over the course of the first weeks the main focus would be to study and learn about steganography technique and learn which places in TCP/IP headers are suitable for hide information.
The next weeks was development a technique to insert and send data in the header and implementation of this technique that we developed.
Then we will check with Wireshark and after send the packet, we will recover the original message.

Last two weeks of the planned time were reserved for writing this documentation and performing various tests over the written software.

4.2. Project development

First of all, for the implementation of steganography we will use a cover channel inside the TCP or IP header.
A covert channel is a computer security attack that creates a capability to transfer data between processes that are not supposed to be allowed to communicate by the computer security police.

To make communication possible it's usually necessary a preliminary agreement between sender and receiver that encodes a message in order for the receiver to be able to interpret it.

The first thing was to study and learn about network steganography techniques and how can they be used on the IPv4 network protocol. Then I started to develop how I can put information inside of the TCP protocol, to do that I needed to have access to the transport layer to manipulate those TCP fields, so the answer that we found for that problem was the use of Scapy.

Scapy is a Python Framework that enables the user to send, sniff, dissect and forge network packets. This capability allows the construction of tools that can prove, scan or attack networks. In other words, Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match request and replies, and much more. Scapy can easily handle most classical tasks like scanning, tracerouting, attacks or network discovery.

So once we checked the documentation of Scapy and know how to use it, we started to develop our steganography technique that relies on the fields of TCP and IP headers that are not checked for sending.

In the [Tab1] we can observe which of the fields of both headers are more suitable for hide information. All the other fields are more complicated to use as a cover channel because are necessary for sending the packets, like IP address or port, etc.

IP header	TCP header
Identification	Window Size
Total Length	TCP Option
TTL(Time To Live)	ACK number

Tab1. Suitable places in headers.

When we have chosen in which fields we can put the information, we start to send packets with the new modified fields, and with the network sniffer we will be capturing these packets and see that we modify the TCP header.

Now we have to export to a text file all these packets for analyzing and processing the file, recover the original message and write it in a new file.
Currently the functionality of the application for sender is:

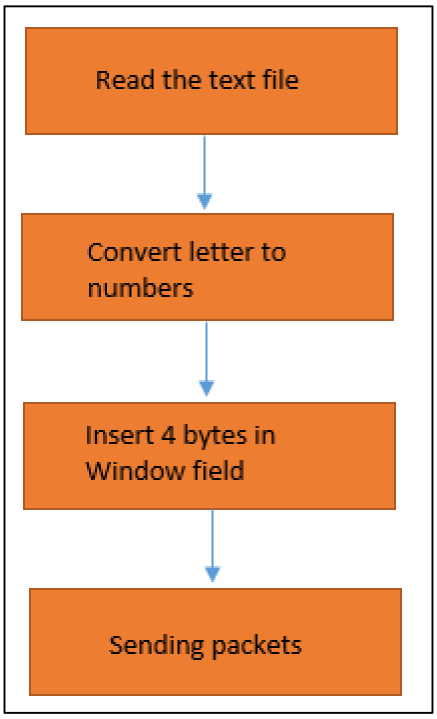


Image 1: Sender functionality

And the current functionality of the application for the receiver is:

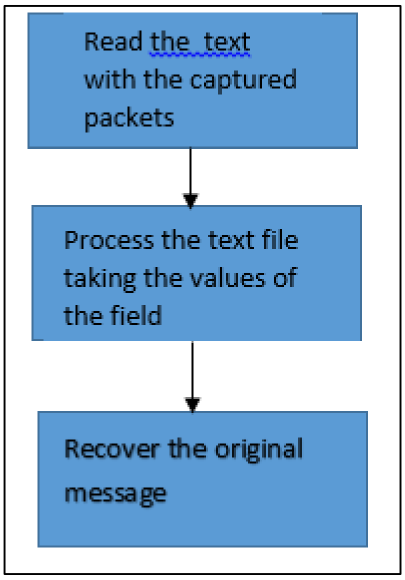

Read the text with the captured packets
Process the text file taking the values of the field
Recover the original message

Image 2: Receiver functionality

5. Implementation

Before we start explaining how to carry out the implementation, specifying libraries, and main classes of our code, we must first clarify why we have chosen Python as our development language.

5.1. Choosing Python as the development language

One of the initial decisions to make was to select an adequate programming language that could fulfill our expectations regarding performance and ease of use.
A first concern was to choose a language capable of working under an object-oriented Paradigm. From this initial condition we pre-selected Java and Python as our languages of choice to develop our application. In an initial research, we thought about using Java as our first development language because we could use a client-server and using the library JnetPcap to manipulate the TCP and IP headers but we had some problem with that library.

The problem that we had with that library is when I created a client-server with Java we use for send and receive the next Java syntax: (*out.writeUTF()* for send and *in.readUTF()* for receive), so when we want to send the packet that we created with that library JNetPcap , we created a TCP packet but the packet is a JMemoryPacket type, and when I wanted to send from the Client to the Server I had an error that is not supported by UTF , so for this reason we changed to Python as our final development language.

We have structured this code in 3 classes:

- Steganography Class
- Conversion Class
- Main Class

5.2. Steganography Class

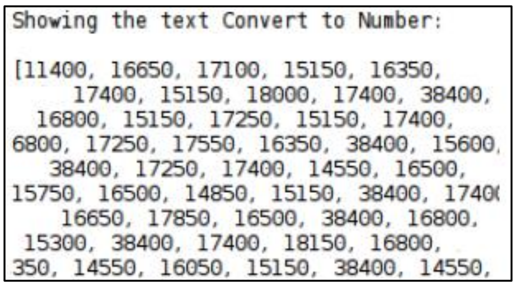
The steganography class is composed of 3 methods that are: OpenFile, Conversition to Integer and Sending File that will be explained below:

- **OpenFile:**

This Method will open the file, read it and store it in an array.

- **Conversion to Integer:**

The after we store the text in an array, it's converted to an integer, because in Scapy, TCP header fields must be filled with numbers, so we convert the letters store in the array with the python syntax *ord()*.
In python the ord(), convert from string into the ASCII value of the letter, after that we modify the original ASCII value , by multiplying by 2 numbers, and the result of this operation will be the same size as one original packet.
In the image below we can see a piece of the text converted into number



```
Showing the text Convert to Number:

[11400, 16650, 17100, 15150, 16350,
     17400, 15150, 18000, 17400, 38400,
  16800, 15150, 17250, 15150, 17400,
6800, 17250, 17550, 16350, 38400, 15600,
   38400, 17250, 17400, 14550, 16500,
15750, 16500, 14850, 15150, 38400, 1740
    16650, 17850, 16500, 38400, 16800,
 15300, 38400, 17400, 18150, 16800,
350, 14550, 16050, 15150, 38400, 14550,
```

Image3: Text into Number

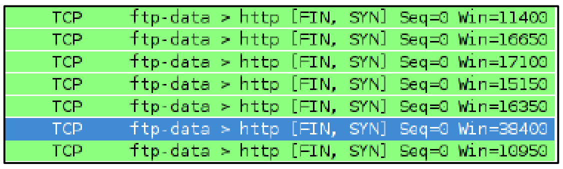
- **Sending Packets:**

After we had the text converted into number and multiplied by these 2 numbers, it's time to insert into the field of TCP header, for this example we will use the field of Window, so that means that the length of this field is 16bits , so we only can put 4 bytes of information per packet.

All the others fields are generate automatically with the defaults values when Scapy create the headers, also another values you have to change are:

a) IP source
b) IP destination
c) Source port
d) Destination port.

So once we insert the data into the windows field and send it, if we captured with Wireshark this packets, we will observe that the length of the window is same or similar to one original packet but in fact are our message inside.



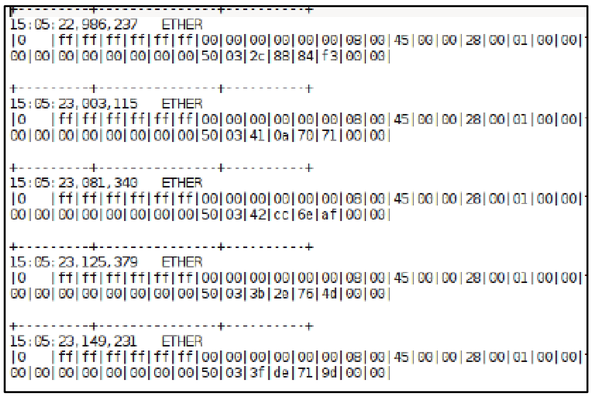
```
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=11400
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=16650
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=17100
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=15150
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=16350
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=38400
TCP     ftp-data > http [FIN, SYN] Seq=0 Win=10950
```

Image 4: Sending packet.

As we can see, we can prove that the length of the Window in Image 2 are same values that in the Image 1.

5.3. Conversion Class

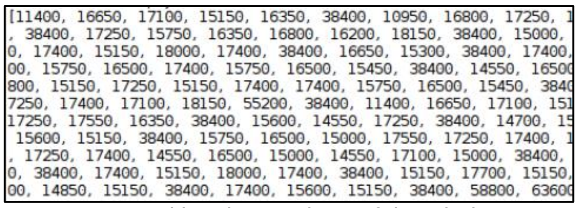
First of all the first thing before call this class is export from Wireshark the captured packets.



```
+---------+---------------+----------+
15:05:22,986,237   ETHER
|0   |ff|ff|ff|ff|ff|ff|00|00|00|00|00|00|08|00|45|00|00|28|00|01|00|00|
00|00|00|00|00|00|00|00|50|03|2c|88|84|f3|00|00|

+---------+---------------+----------+
15:05:23,003,115   ETHER
|0   |ff|ff|ff|ff|ff|ff|00|00|00|00|00|00|08|00|45|00|00|28|00|01|00|00|
00|00|00|00|00|00|00|00|50|03|41|0a|70|71|00|00|

+---------+---------------+----------+
15:05:23,081,340   ETHER
|0   |ff|ff|ff|ff|ff|ff|00|00|00|00|00|00|08|00|45|00|00|28|00|01|00|00|
00|00|00|00|00|00|00|00|50|03|42|cc|6e|af|00|00|

+---------+---------------+----------+
15:05:23.125.379   ETHER
|0   |ff|ff|ff|ff|ff|ff|00|00|00|00|00|00|08|00|45|00|00|28|00|01|00|00|
00|00|00|00|00|00|00|00|50|03|3b|2a|76|4d|00|00|

+---------+---------------+----------+
15:05:23,149,231   ETHER
|0   |ff|ff|ff|ff|ff|ff|00|00|00|00|00|00|08|00|45|00|00|28|00|01|00|00|
00|00|00|00|00|00|00|00|50|03|3f|de|71|9d|00|00|
```

Image 5: Export the captured packets.

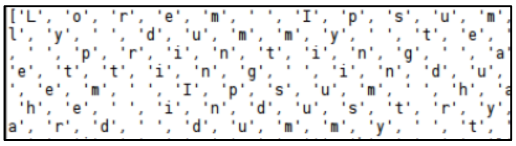
After we done the exportation of the packets [Image 3], this class will read this text file and delete all the symbols like (+,- and |) and only select the numbers that are in our Window field.



```
[11400, 16650, 17100, 15150, 16350, 38400, 10950, 16800, 17250, 1
, 38400, 17250, 15750, 16350, 16800, 16200, 18150, 38400, 15000,
0, 17400, 15150, 18000, 17400, 38400, 16650, 15300, 38400, 17400,
00, 15750, 16500, 17400, 15750, 16500, 15450, 38400, 14550, 1650
800, 15150, 17250, 15150, 17400, 17400, 15750, 16500, 15450, 384
7250, 17400, 17100, 18150, 55200, 38400, 11400, 16650, 17100, 151
17250, 17550, 16350, 38400, 15600, 14550, 17250, 38400, 14700, 15
 15600, 15150, 38400, 15750, 16500, 15000, 17550, 17250, 17400, 1
, 17250, 17400, 14550, 16500, 15000, 14550, 17100, 15000, 38400,
0, 38400, 17400, 15150, 18000, 17400, 38400, 15150, 17700, 15150,
00, 14850, 15150, 38400, 17400, 15600, 15150, 38400, 58800, 6360
```

Image 6: Taking the numbers of the window

After we selected the numbers of our TCP header [Image 4], we will have to divide per our 2 numbers that we multiply at the beginning and the result will be our original text [Image 5].



```
['L', 'o', 'r', 'e', 'm', ' ', 'I', 'p', 's', 'u', 'm',
l', 'y', ' ', 'd', 'u', 'm', 'm', 'y', ' ', 't', 'e',
, ' ', 'p', 'r', 'i', 'n', 't', 'i', 'n', 'g', ' ', 'a
'e', 't', 't', 'i', 'n', 'g', ' ', 'i', 'n', 'd', 'u',
', 'e', 'm', ' ', 'I', 'p', 's', 'u', 'm', ' ', 'h', 'a
 'h', 'e', ' ', 'i', 'n', 'd', 'u', 's', 't', 'r', 'y',
a', 'r', 'd', ' ', 'd', 'u', 'm', 'm', 'y', ' ', 't',
```

Image 7: Convert to letter.

We can observe that our original message it was a random Lorep Ipsum sentences.

5.4. Main Class

The Main Class will call the other two classes and will interact with the user for sending and receive message.

6. Conclusions

Overall, the project has been completed satisfactorily on the scheduled time. We have successfully developed an application capable of performing the designed tasks accomplishing our requirements and fulfilling the objectives of the project.
We were surprised to notice how a well-performed and extensive design stage could lead to having way less trouble when developing the application. The large time designated to plan the application really payed off when the moment to write it down in Python code came. Another positive conclusion from doing this project has been observing the vast amounts of Python information and documentation available on the Internet.

Every time the development got stuck by something, it was extremely easy to find out information on multiple websites about how to solve the problem. However, this project can be used as a proof that it's possible using Python with Scapy, manipulate the header of the TCP/IP protocols and send them with the information inside the header.
Also using the steganography in the TCP and IP headers we can avoid during the sending message if someone are trying to do a Man In The Middle Attack (MITM), and don't know that our information is in the headers never will know

our message and even they could know that are in the header if we use also some cryptography will be really complicated for our enemy knowing the message.
Also the variety of steganographic carriers is constantly increasing.

7. Future work

Considering the last version of the application, there is still a lot of room for future improvement. In this section we will discuss some of these features that didn't make it to the final version due schedule limitations.

- We can implemented some cryptography algorithm before putting the data into the header for more security (e.g. some public algorithm as RSA).

- In this project about the Network Steganography, we treated the steganography focusing in one determinated way, but concerning about Steganography there a lot of differents and possibles way to treat this topics.

- Investigate how steganography works on botnets and look whether the current technique that uses by botnets can be improved when the botnets ahve to communicate with the Central Computer or Brain for send the stolen data.

- Investigate other networks techniques for sending information in other layers and find more covert channels.

7.1. Modern usage of steganography

Some moders usage of the steganography can be found in Malware or computer games that means that is a topic that is in constaintly growing in more places.

a) **Malware**: Computer worms and others forms of malware are presently exploiting steganography for the purpose of obtaining functioning command and transfering hijacked data.
It seems thats steganography is starting to be popular with Android malware.

b) Computer games: It's suspected that multiplayer games may be a good cover for covert communication. Playstation and Xbox communication is free of legal invilation from beginning 2012.

c) Null cipher: It's a normal text written in the clear, but includes a hidden message. For example:
"Fishing freshwater bends and saltwater coast reward anyone feeling stressed.Resourceful anglers usually find masterful leapers fun and admit swordfish rank overwhelming anyday"
If we take out the third letter in each word , we get. "Send Lawyer,Guns and Money".

8. Acknowledgement

I would like to thank my tutor of this work, Guillerno Navarro(UAB-dEIC) for his help and support during the performance of this work.

9. References

[1] Joff Thyer (2011), Covert channels using IP packet Header [Online] Available; http://www.irongeek.com/videos/derbycon1/tcpip-header-covert-channels.pdf

[2] Philippe Biondi (April 19, 2010) Scapy documentation [Online] Available: http://www.secdev.org/projects/scapy/doc/usage.html

[3] Python Software Foundation(Last Update,2014) Python documentation, [Online] Available: https://docs.python.org/2/tutorial/

[4] Himanshu Arora (March 26,2012) IP Protocol Header Fundamentals Explained with Diagrams [Online] Available: http://www.thegeekstuff.com/2012/03/ip-protocol-header/

[5] Krzysztof Szczypiorski (November 4th,2003) Steganography in TCP/IPNetworks state of the art and a proposal of a new system-HICCUPS [Online] Available: http://www.tele.pw.edu.pl/~krzysiek/pdf/steg-seminar-2003.pdf

[6] Steven J. Murdoch and Stephen Lewis (July 29, 2005), Embedding covert channel into TCP/IP [Online] vailable:http://www.cl.cam.ac.uk/~sjm217/papers/ih05coverttcp.pdf

[7] J.R.Krenn, "Steganography and Steganalysis", January 2004

[8] Network Steganography and anomaly detection http://stegano.net

[9] [Online], Available, http://en.wikipedia.org/wiki/Steganography

[10] Józef Lubacz, Wojciech Mazurczyk, Krzysztof Szczypiorski Principles and Overview of Network Steganography

[11] Elzbieta Zielinksa, Wojciech Mazurczyk and Krysztof Szxzypiorski, March 2014 , Trends In Steganography.